Influence of Encoding and Neighborhood in Landscape Analysis and Tabu Search Performance for Job Shop Scheduling Problem

Israël Tsogbetse[a], Julien Bernard[b], Hervé Manier[a], Marie-Ange Manier[a]

[a] *UTBM, CNRS, FEMTO-ST, 90010 Belfort, France*
[b] *Université de Franche-Comté, CNRS, FEMTO-ST, 25000 Besançon, France*

Abstract

Fitness landscape analysis is used to understand search spaces of combinatorial problems that can hardly be solved exactly, such as the job shop scheduling problem. We analyze the influence of the solution encodings and the neighborhood operators on usual metrics for landscape analysis, and try to correlate the results to the performance of a local search method such as tabu search using these encodings and operators for a wide range of instances of the job shop scheduling problem.

Keywords: Scheduling, Metaheuristics, Evolutionary computations

1. Introduction

Metaheuristics are used when exact methods are limited by the size of the problem and cannot find a solution in a reasonable amount of time. In this case, the choice of the metaheuristic algorithm and its features can improve the discovery of good solutions in the search space, especially the solution encoding and its neighborhood operator in a local search metaheuristic. Fitness landscape analysis aims at understanding the search spaces through various properties and their related metrics. The goal is to explain why a metaheuristic performs well (or poorly), hopefully before computing a solution with the chosen metaheuristic.

The job shop scheduling problem (JSP) is a good candidate for fitness landscape analysis as it is a well known combinatorial problem that is hard to

Email addresses: `israel.tsogbetse@utbm.fr` (Israël Tsogbetse), `julien.bernard@femto-st.fr` (Julien Bernard), `herve.manier@utbm.fr` (Hervé Manier), `marie-ange.manier@utbm.fr` (Marie-Ange Manier)

solve, even for small instances. Metaheuristics that are used to solve the job shop scheduling problem are generally based on the choice of a solution encoding and its related neighborhood operator to navigate through the search space. The goal of this paper is to be able to make recommendations of encodings and their neighborhood operators that ensure good performance for a given metaheuristic, hopefully based on the results given by the landscape analysis.

Through fitness landscape analysis, we have computed different metrics for a set of well-known encodings and neighborhood operators. Then, we compared these results to the performance of a tabu search using these encodings and neighborhood operators. The experiments have been made on a very wide range of instances of the problem from the literature. We tried to correlate the landscape analysis with the results of the metaheuristic in order to see if the analysis can be transformed into a direct recommendation.

In section 2, we give an overview of the analysis of encodings and neighborhood operators for the JSP in the literature, in particular the cases of fitness landscape analysis. In section 3, we give a formal definition of the job shop scheduling problem and we describe the different encodings and neighborhoods, as well as a generic tabu search algorithm and the tested instances. In section 4, we present and analyze the results of our experiments.

2. Related works

In this section, we first analyze some previous works about encodings or neighborhood operators comparisons. Then we look at related works about fitness landscape analysis for the job shop scheduling problem.

Cheng et al. (1996) made a list of nine different encodings used in genetic algorithms solving the JSP. Several properties were studied such as the complexity of the decoding process or the memory requirements. Ponnambalam et al. (2001) extended this study on four of the encodings comparing the minimum makespan. The job permutation encoding (later called `job`) provided the best results. Abdelmaguid (2010) also extended this study on six of the encodings comparing the average optimality gap. The encoding based on machines had the minimum gaps. Finally Jorapur et al. (2014), on the same set of encodings, stated that a second encoding based on jobs had the best performance. These contradictory conclusions can be explained by the instances that were used, respectively 20 square instances, 40 square and rectangular instances and 68 square and rectangular instances. Moreover, in each case, the choice of the operators for selection, crossover and mutation used with each encoding was absent from the protocol descriptions. In

Şahman & Korkmaz (2022), a comparison of three encoding schemes was carried out for the first time with a custom metaheuristic to solve JSP. The best results in terms of makespan value were obtained by the Smallest Position Value (SPV) encoding. Overall, these studies appear to have divergent results due to the various experimental conditions (instances or associations with operators).

Besides JSP, comparisons of encodings have been done for other scheduling problems such as flow shop (Fernandez-Viagas et al. (2019)), open shop (Tsujimura et al. (1997)), or unrelated machines (Durasević & Jakobović (2016); Vlašic et al. (2020)). Like previous works, they concentrate on both the quality of the produced schedules and the properties of the encodings.

Regarding neighborhood operators, most papers in the literature use either a swap operator (later called `swp`) or a variant of the critical adjacent swap operator introduced by van Laarhoven et al. (1992) (later called `cas`) without questioning their relevance. Some comparative studies exist to evaluate the performance of neighborhood operators. For solving a variant of the JSP, Mastrolilli & Gambardella (2000) compare two operators based on the insertion of operation on the critical path using a graph for the representation of the solution. Kuhpfahl & Bierwirth (2016) used a disjunctive graph to compare twelve operators which are combinations and variants of critical swap and insertion operators. No dominant operator has been identified. In both cases, the operators are tested on a single encoding.

We see that few works focus on the impact of encodings and neighborhood operators on the performance of metaheuristics for solving the job shop problem, in particular their possible combinations. Moreover, there is room for improvement in the experimental conditions (especially number and diversity of instances). We try to address these shortcomings in the following sections. Some other works focus on fitness landscape analysis for the JSP in order to better understand the performance of metaheuristics.

Landscape differences between some hard and easy JSP benchmark instances have been investigated by Mattfeld et al. (1999) using fitness distances, distributions in terms of entropy, smoothness and correlation length. The landscapes have been generated using local search solving methods. They concluded that landscape structure implies, on one hand, that smoothness is necessary for neighborhood operators to work properly. Neighborhood search is more favorable for hard JSP instances than adaptive search even though genuine neighborhood search is partially hindered by plane surfaces of landscapes. Bierwirth et al. (2004) analyzed the landscape generated by the Fisher and Thompson 10×10 JSP instances (Fisher & Thompson (1963)) using a disjunctive graph representation and adjacent-swap neighborhood.

They showed that the connectivity of solutions is variable along the space and this makes the landscape irregular. So, random walks on such landscape will be biased. However they affirm that, in JSP, high-quality solutions generally possess high degrees of connectivity. So they conjecture that such irregularity should aid random walks and stochastic local search algorithms for the JSP, guiding search toward regions of the fitness landscape containing high-quality solutions. In Streeter & Smith (2006), the impact of the ratio of jobs to machines on the landscape of random instances for the JSP has been studied. It has been shown, among other things, that for low ratio values, low makespan schedules are clustered in a small region of the search space and these schedules possess many common attributes. This is not the case for high ratio values.

3. Problem description, search space setups and metrics

3.1. Problem Description

The job shop scheduling problem consists in scheduling n jobs $(J_i)_{1\leq i\leq n}$ on m machines $(M_k)_{1\leq k\leq m}$. Each job J_i has N_i operations $(O_{ij})_{1\leq j\leq N_i}$ that must be scheduled in that order. $N = \sum_{i=1}^{n} N_i$ is the total number of operations. Each operation O_{ij} has a dedicated machine μ_{ij} and a processing time p_{ij}. A machine can execute at most one operation at a time, and operations cannot be preempted. The goal is to minimize $C_{\max}$, the maximum completion time of all the jobs, also called makespan. Using the three field notation introduced by Graham et al. (1979), our problem is $J \mid\mid C_{\max}$ (J denoting job shop and the empty middle denoting no additional constraint).

3.2. Encodings

In this section, we describe four encodings: `job`, `ope`, `mch` and `tim`, as they are suitable and the most often used for a job shop scheduling problem. The first three encodings are permutation encodings. For these encodings, the schedule is computed greedily as follows: for each operation in the order given by the encoding, the starting date is the minimum between the end date of the last operation on the considered machine and the end date of the previous operation of the same job. If the solution is feasible, this procedure ensures a semi-active schedule.

`job` is a job list encoding introduced by Bierwirth (1995). A solution is an ordered list of N job numbers, in which each job J_i appears N_i times. This encoding covers the space of all possible active schedules. Two different solutions can lead to the same schedule. All the solutions are feasible. `ope`

is an operation list encoding. A solution is an ordered list of N operations. Each operation appears exactly once in the list so the list is a permutation of all the operations. A solution may be infeasible if an operation is before one of its predecessors in the list. `mch` is a machine list encoding used by van Laarhoven et al. (1992) for the same problem. A solution is a list of m ordered lists of operations that are assigned to each machine. The size of the lists varies according to the number of operations for each machine. A solution may be infeasible when two operations from two different jobs are interlocked on two different machines while they are waiting for a predecessor that should be processed later in the other list.

`tim` is a original time list encoding that, to our knowledge, has never been proposed. A solution is an ordered list of N times. Each time corresponds to an operation and represents the minimum delay between the start of the operation and the end of the previous operation of the same job (or the start of the schedule for the first operation). As the delay is a minimum, there is always a valid schedule that satisfies the delay constraints in the encoding. But the schedule may not be semi-active, so we compute an equivalent semi-active schedule. Contrary to the other encodings, `tim` is not a permutation encoding.

3.3. Neighborhood

In this section, we describe the different neighborhood operators used in conjunction with the previous encodings. Not all operators apply to each encoding. Most of these operators are commonly found in the literature for solving job shop problems with metaheuristics.

`swp` is a swap operator, it exchanges two randomly chosen elements in a solution. This operator can be applied to `job` (and the result should be different from the original solution), `ope` and `mch`. In the case of `mch`, the operator is applied to one randomly chosen operation list. `ins` is an insertion operator, it removes a randomly chosen element from a solution and inserts it at a randomly chosen different index. The operator can be applied to `job`, `ope` and `mch` (with the same adaptation as before). `rev` is a reverse operator, it randomly chooses two different indices in a solution and reverses all the elements between those two indices. The operator is appropriate for `job`, `ope` and `mch` (with the same adaptation as before). `adj` is an adjacent swap operator, it exchanges two randomly chosen adjacent elements in a solution. It is a special case of the three previous operators. This operator is suitable for `job`, `ope` and `mch` (with the same adaptation as before). These first four neighborhood operators apply to any permutation encodings.

`cas` is a critical adjacent swap operator introduced by van Laarhoven et al. (1992). This operator exchanges two adjacent operations in a critical block, i.e. two adjacent operations that are on a same machine and on a critical path. This operator ensures that if a current solution is feasible, then its neighbors are also feasible. This operator is applied to `ope` and to `mch`, and not `job` as it does not reference any operation explicitly.

`tim` is a time change operator, it is created specifically for the `tim` encoding. A subset of the elements is modified (15% in our case) as follows: an element whose value t is zero is modified with a new value chosen randomly with a negative binomial distribution with parameters $r = 4 \times \max(1, t)$, $p = 0.8$. If t is not zero, this distribution ensures an average value equal to t but a median value less than the average: the value of the elements tends to decrease (which is expected to obtain a better schedule) but sometimes it increases a lot (to reach very different schedules). This neighborhood operator cannot be applied to a permutation encoding.

3.4. Properties and Metrics

In fitness landscape analysis, properties and their associated metrics are used to describe the landscape in order to better understand the structure of the search space. The following ones are considered in our study.

Neutrality gives an indication on neighbors that have the same fitness, called neutral neighbors (Tari et al. (2021)). A neutral landscape is characterized by a significant proportion of neutral neighbors and implies the presence of plateaus and ridges in the landscape. The metric that is used traditionally to measure neutrality is the *neutrality rate*. It is the average proportion of neutral neighbors in the landscape. As the landscape is assumed to be statistically isotropic, it can be measured with a random walk in the landscape.

Ruggedness measures the distribution of local optima and the size of their attraction basins. A rough landscape has neighbors with very different fitness values whereas a smooth landscape has neighbors with almost the same values. A common metric to measure ruggedness is the *correlation length* (technique 2 in Malan & Engelbrecht (2013)) that is the distance τ beyond which the majority of solutions becomes uncorrelated. τ is derived from the auto-correlation function $\rho(n)$ which measures the correlation between a solution and the n-th subsequent solution in a random walk: $\tau = -\frac{1}{\ln|\rho(1)|}$. A high value of τ indicates a smooth landscape.

Evolvability is the capacity to evolve in the landscape toward better fitness. It is also called searchability. A metaheuristic may take advantage

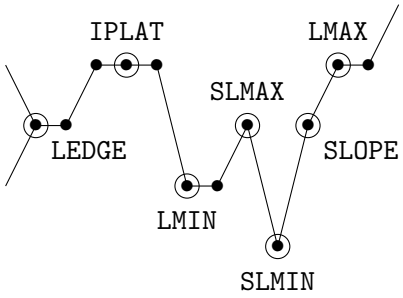

Figure 1: Position types (Hoos & Stützle (2005)): `IPLAT` (interior plateau), `LEDGE` (local edge), `LMAX` (local maximum), `LMIN` (local minimum), `SLMAX` (strictly local maximum), `SLMIN` (strictly local minimum), `SLOPE` (slope). For example, `LMIN` point only has neighbors with the same fitness or higher fitness.

of a landscape with high evolvability to get better solutions faster. Evolvability can be measured with *accumulated escape probability* (technique 22 in Malan & Engelbrecht (2013)). For each solution of a representative set, the proportion of neighbors with improved fitness in the landscape is computed. The accumulated escape probability is the average value of all these values.

Position type distribution is a measure that classifies a solution according to the fitness differences observed in its neighborhood (Hoos & Stützle (2005)). This gives seven position types that are summarized in figure 1. This metric is linked to the previous properties, e.g. a high proportion of `IPLAT` solutions denotes a neutral landscape.

Many other metrics have been proposed in Malan & Engelbrecht (2013) and later in Malan (2021). Most of them do not apply to our problem, either because they need a binary encoding or because they require the knowledge of properties that are difficult to compute for our problem (optima network, global optima, "distance" between two solutions).

3.5. Instances

We used two sets of instances of job shop scheduling problems, representing 2742 instances, to test our 15 encoding-neighborhood pairs. The first set is composed of 242 *classical instances* that are described extensively by Hoorn (2018). Table 1 gives the main characteristics of these instances. The second set includes 2500 *generated instances* that have been created by Strassl & Musliu (2022) with $n = 10, 20, \ldots, 100$, and $m = 10, 20, \ldots, 100$. Different distributions have been used for generating the processing times of the operations: `gen-const` for constant distribution equals to 1; `gen-uniform-99` for uniform distribution in $[1, 99]$; `gen-uniform-200`

Prefix	Count	n	m	Ref
`abz`	5	10,20	10,15	Adams et al. (1988)
`dmu`	80	20,30,40,50	15,20	Demirkol et al. (1998)
`ft`	3	6,10,20	5,6,10	Fisher (1963)
`la`	40	10,15,20,30	5,10,15	Lawrence (1984)
`orb`	10	10	10	Applegate & Cook (1991)
`swv`	20	20,50	10,15	Storer et al. (1992)
`ta`	80	15,20,30,50,100	15,20	Taillard (1993)
`yn`	4	20	20	Yamada & Nakano (1992)

Table 1: Characteristics of Classical Instances

for uniform distribution in $[1, 200]$; `gen-binom` for binomial distribution with $n = 98$ and $p = 0.5$; `gen-nbinom` for the negative binomial distribution with $r = 1$ and $p = 0.5$. We notice that in all theses instances, $N_i = m$. These instances have been used originally to compare several algorithms for solving the job shop problem. As they are recent, they are not widely studied but have the advantage ot covering a large range of n and m.

3.6. Tabu Search

In order to test the different encodings and neighborhood operators and find correlations with the results of landscape analysis, we implemented a tabu search adapted from Taillard (1994) for the same problem. The adaptation consists in replacing the encoding and neighborhood operator (`mch-cas` in the original implementation) by one of our encoding-neighborhood pairs. We preserved the length of the tabu list (called L in Taillard (1994)).

We arbitrarily chose to limit the exploration to 20 neighbors (feasible or not), and to limit the whole search to 10 seconds. If the algorithm cannot find a candidate (for example, no neighbor is feasible), then the algorithm restarts from the beginning. This may limit its performance when the operators have difficulties to find a feasible neighbor.

4. Experiments and Analysis

To compute the four considered metrics, we made three kinds of experiments for each encoding and its neighborhood operators, and each instance:

1. In the first experiment, we generate 20 random walks of length 1000 from a feasible solution. For each step of the walk, the first found feasible neighbor is chosen as the next step. We assume the landscape

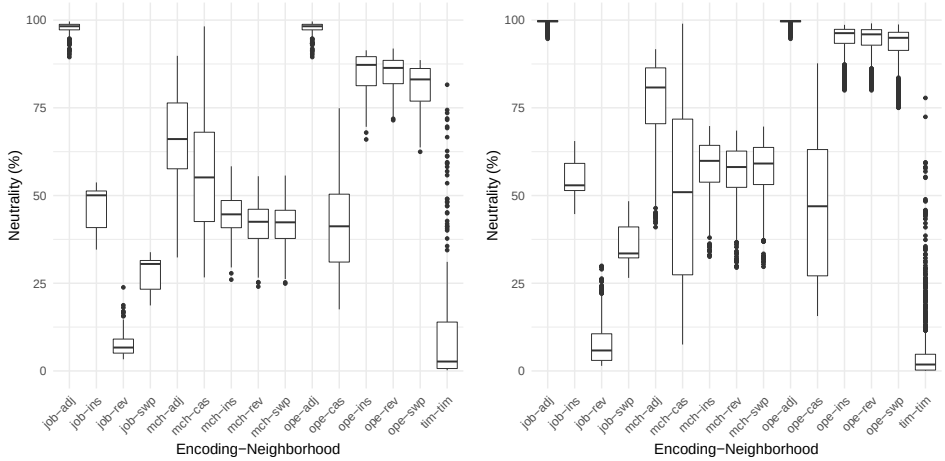

Figure 2: Neutrality Rate for Classical Instances (left) and Generated Instances (right)

is statistically isotropic, i.e. the starting position does not play any role in the result. From this experiment, we compute the neutrality rate and the correlation length.

2. In the second experiment, we uniformly generate 1000 feasible solutions and for each solution, we generate 100 feasible neighbors. Generating solutions with a discrete uniform distribution is only possible for `job` and `ope` (because a feasible solution in `ope` is in bijection with a solution in `job`). From this experiment, we compute the accumulated escape probability and the position types distribution.
3. In the third experiment, we compute the results of 10 runs of the tabu search and keep the best one. For each instance, the results are ranked, with ties having the same rank.

These three experiments utilized several months-CPU and generated a total of 17 GB of raw text data.

4.1. Neutrality Rate

Figure 2 shows the neutrality rate for classical instances and generated instances respectively.

The neutrality rate is quite similar for classical and generated instances. It essentially depends on the encoding-neighborhood pair. Two pairs have a very high neutrality, close to 100%: `job-adj` and `ope-adj`. Exchanging two elements in these solutions does not change the schedule or provide a schedule with the same makespan, hence the very high neutrality.

Three other pairs have high neutrality, between 80% and 95%: `ope-ins`, `ope-rev` and `ope-swp`. It comes from the difficulty to find feasible neighbors with the `ope` encoding. Most neighbors are not so different and thus give quite similar scheduling with similar fitness.

A large group of pairs have medium neutrality, between 25% and 75%: `job-ins`, `job-swp`, `mch-adj`, `mch-cas`, `mch-ins`, `mch-rev`, `mch-swp`, `ope-cas`. Similar to the `ope` encoding, the `mch` encoding with `ins`, `rev` and `swp` neighborhood have similar profiles due to the fact that it is difficult to find feasible neighbors. Again, the `adj` operator (which is in the intersection of the three others) gives a higher neutrality.

Finally, two pairs have low or very-low neutrality: `job-rev` and `tim-tim`. In the case of `job-rev`, reversing many elements in the solution can give a totally different solution with a totally different schedule and makespan. As for `tim-tim`, the neighborhood operator changes enough elements in the solution so that the makespan is different.

4.2. Correlation Length

Figure 3 shows the correlation length for classical and generated instances. The correlation length is plotted with respect to n, the number of jobs, and a logarithmic y scale.

The correlation length varies from 2 to more than several thousands, but stays within the same order of magnitude for a single encoding-neighborhood pair. In the vast majority of the cases, the correlation length increases with n. This is due to the nature of the neighborhood operators that have a smaller relative impact on the solutions when n (and m) increases, which gives neighbor solutions with more correlation. On the contrary, the `tim` neighborhood operator modifies a proportion of the solution and the correlation length decreases a little.

The `job-rev` pair has a correlation length between 2 and 3, meaning that neighbors with a distance greater than 3 are uncorrelated. In fact, applying `rev` twice successively on a `job` encoding is like getting a random solution.

Some pairs show average correlation lengths, from 10 to 80: `job-ins`, `job-swp`, `ope-ins`, `ope-rev`, `ope-swp`. The landscape is neither too rough, nor too smooth. The operators modify the solutions enough, but not too much so that the correlation length is what we could expect in order to avoid too many local optima, while having uncorrelated solutions quite quickly.

Another group exhibits large correlation lengths, from 80 to 200, which means a smooth landscape: `mch-cas`, `mch-ins`, `mch-rev`, `mch-swp`, `ope-cas`. The `cas` operator restricts the number of possible neighbors and all neighbors do not differ much from the origin and so, are quite correlated even after

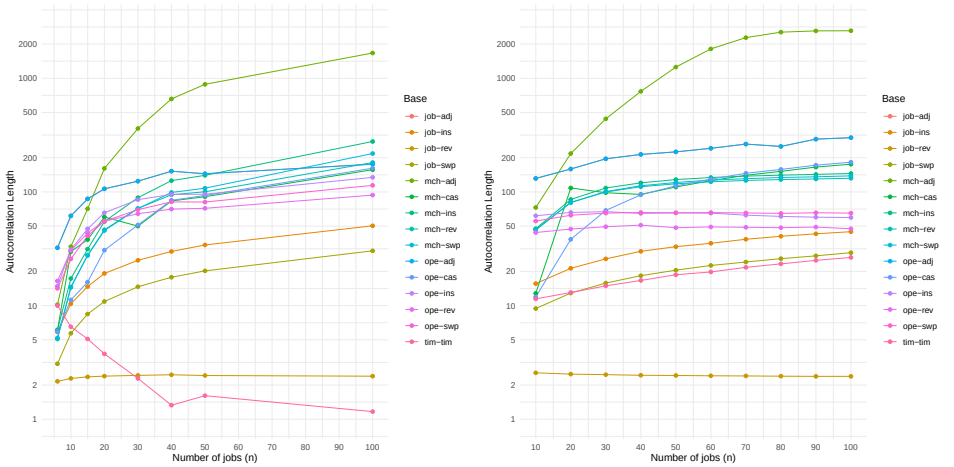

Figure 3: Correlation Length for Classical Instances (left) and Generated Instances (right)

many steps. The `mch` encoding suffers from the same pitfall, the non-`cas` neighborhoods operators only modify the operations on a single machine. It is not enough to find uncorrelated solutions quickly.

The `adj` neighborhood operator gives very high correlation lengths of several thousands. As stated before, it does not modify the solution enough, so that a large number of steps is necessary to obtain an uncorrelated solution.

Finally, the `tim-tim` pair gives a very spread repartition of correlation length in the case of generated instances (not shown in the figure). We observed that the correlation length depends on the probability distribution that is used to generate the processing times. Two groups of instances clearly appear: one group with medium correlation length (20-50) for `gen-const` and `gen-nbinom`; and a group with low correlation length (5-20) for `gen-binom`, `gen-uniform-200` and `gen-uniform-99`. This is consistent with the results obtained with classical instances where the processing times are generated uniformly.

4.3. Accumulated Escape Probability

Figure 4 shows the accumulated escape probability for classical and generated instances respectively.

`job-adj` and `ope-adj` have a very low escape probability. This is linked to the very high neutrality rate of their landscape: it is difficult to find a better neighbor among neighbors with the same fitness. Similarly, `ope-ins`, `ope-rev` and `ope-swp` have a low escape probability because they have a

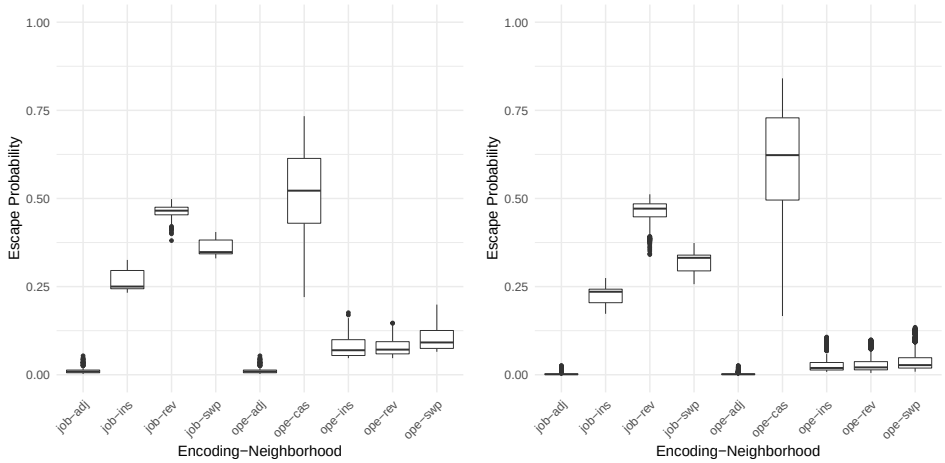

Figure 4: Accumulated Escape Probability for Classical Instances (left) and Generated Instances (right)

high neutrality rate. `job-ins` and `job-swp` have medium escape probability, which is again very related to their medium neutrality rate.

Overall, for this problem, on average, half of non-neutral neighbors are better neighbors, whatever the encoding-neighborhood pair. It does not bring a valuable information for choosing a good encoding-neighborhood pair. It just means that statistically the pairs show predictable behaviors.

4.4. Position Type Distribution

Table 2 shows the position type distribution for classical and generated instances. For these experiments, for each solution, we separate the neighbors in three categories: neighbors that have a better fitness than the solution, neighbors that have a worse fitness than the solution and neighbors that have the same fitness as the solution. Each group is considered to be significant if it has more than 10% of the neighbors. Then from the resulting groups, we derive the position type. For example, if a solution has a significant group of neighbors with the same fitness, and no other significant group, it is categorized as an `IPLAT` solution.

Unsurprisingly, `job-adj` and `ope-adj` have a very high rate of `IPLAT` solutions because their landscape is very neutral. They do not even show any position types where equal fitness is absent (`SLMAX`, `SLOPE`, `SLMIN`). `ope-cas` has an asymmetric profile, with many `LMIN` and a little `SLMIN` but very little `LMAX` and even less `SLMAX` solutions. The rest mainly consists in `LEDGE` solutions. `ope-ins`, `ope-rev` and `ope-swp` share a similar profile, with a high

	SLMAX	LMAX	LEDGE	SLOPE	IPLAT	LMIN	SLMIN
job-adj	-/-	0.3/0.0	0.0/-	-/-	99.3/100.0	0.3/0.0	-/-
job-ins	-/-	1.1/3.5	97.4/93.1	-/-	-/0.0	1.4/3.4	-/-
job-rev	0.9/0.7	0.5/0.5	19.7/22.9	77.7/74.9	-/-	0.4/0.3	0.8/0.6
job-swp	-/-	0.4/0.9	99.1/98.5	0.3/0.0	-/0.0	0.2/0.6	0.0/-
ope-adj	-/-	0.3/0.0	0.0/-	-/-	99.3/100.0	0.3/0.0	-/-
ope-cas	-/0.0	0.1/0.1	37.0/11.9	3.0/1.8	0.1/0.2	58.0/79.8	1.9/6.1
ope-ins	-/-	14.2/2.2	10.9/0.5	-/-	61.3/95.2	13.7/2.2	-/-
ope-rev	-/-	16.3/2.7	7.5/0.6	-/-	60.0/94.1	16.2/2.6	-/-
ope-swp	-/-	19.6/3.9	24.5/2.7	-/-	36.4/89.5	19.6/3.9	-/-

Table 2: Position Type Distribution in percent for Classical/Generated Instances

rate of `IPLAT` due to their high neutrality, but also some noticeable rates of `LMIN`, `LEDGE` and `LMAX` solutions. `job-ins` and `job-swp` also share a similar profile with a very high rate of `LEDGE`, and very little rate of `LMIN` and `LMAX`. Finally, `job-rev` shows a very high rate of `SLOPE` solutions, and a high rate of `LEDGE` solutions. It confirms the very rough nature of its landscape.

4.5. Synthesis on Landscape Analysis

At this point, it is difficult to make predictions on the performance of each encoding-neighborhood pair in the implementation of the tabu search described above just looking at the landscape analysis. For a fixed encoding, which neighborhood operator is the best? For a fixed neighborhood operator, which encoding gives the best results? Table 3 provides a synthesis of the results obtained so far with landscape analysis metrics.

A partial conclusion is that the landscape mainly depends on the encoding-neighborhood pair. The job shop instance plays a very minor role, for some metrics, and more precisely the generation method of the instance may play a role, if any. Further measures (not shown here) show that classical instances are sometimes grouped by family when the results of a metric is more spread than usual. Overall the encoding-neighborhood factor is largely dominating in the results. In the next section, we measure the performance of each pair to try to find a correlation with the landscape analysis measures.

4.6. Performance of Tabu Search

Table 4 and 5 show the ranks obtained by each encoding with one of its neighborhood operator in the tabu search for all the classical instances and generated instances respectively. Rank 1 corresponds to the pairs obtaining the best $C_{\max}$. For example, in table 4, `job-ins` obtained the first place 211 times out of 242 instances. The tables also show the mean rank and the median rank for each pair. For classical instances, we give the number of times the search gave the known optimal bound for each pair.

	Neutrality	Ruggedness	Evolvability	Position Type
job-adj	Max	Low	Min	IPLAT
job-ins	Medium	Medium	Medium	LEDGE
job-rev	Low	Max	High	SLOPE
job-swp	Medium-Low	Medium	Medium	LEDGE
mch-adj	Medium-High	Very Low	-	-
mch-cas	Medium	Medium-Low	-	-
mch-ins	Medium	Low	-	-
mch-rev	Medium	Low	-	-
mch-swp	Medium	Low	-	-
ope-adj	Max	Low	Min	IPLAT
ope-cas	Medium	Low	High	LMIN
ope-ins	High	Medium	Low	IPLAT
ope-rev	High	Medium	Low	IPLAT
ope-swp	High	Medium	Low	IPLAT
tim-tim	Min	Medium or Low	-	-

Table 3: Synthesis on Landscape Analysis

In both cases, job-ins is the best encoding-neighborhood pair in terms of mean rank. It is slightly better in the case of classical instances with a mean rank of 1.19, compared to generated instances, with a mean rank of 2.59. As a reminder, job-ins has a median neutrality rate of 50%, a median correlation length of 20-50, a median escape probability of 25% and is characterized by a very high rate of LEDGE solutions. The job encoding is an encoding with many solutions that lead to the same schedule. Its main property is that all solutions are feasible. The ins neighborhood operator changes the relative order of a single element. In fact, the FLA metrics provide average values, which do not make this pair noticeable among others. This is surprising as there seems to be no correlation with the excellent performance of job-ins on all instances.

In the case of classical instances, excluding job-ins, three group appear. A first group of four pairs with mean ranks between 3.11 and 3.81: mch-adj, ope-ins, job-swp and ope-swp. Then a second group of five pairs with mean ranks between 6.33 and 8.99: tim-tim, mch-cas, ope-rev, ope-cas, job-rev. Finally, a last group of five pairs with mean ranks between 10.81 and 13.36: mch-ins, ope-adj, job-adj, mch-swp and mch-rev.

In the case of generated instances, the groups are slightly different. First a group of four pairs with mean ranks between 3.06 and 3.74: mch-cas, tim-tim, ope-cas and job-swp. Then a second group of five pairs with mean ranks between 6.05 and 9.35: ope-ins, ope-swp, mch-adj, job-rev, ope-rev. Finally, a last group of five pairs with mean rank between 12.31 and 13.66: mch-ins, mch-rev, mch-swp, job-adj and ope-adj.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Mean	Median	OPT
job-ins	211	25	2	2	1		1									1.19	1.00	36
mch-adj	49	62	47	35	29	7		11	1					1		3.11	3.00	35
ope-ins	26	56	60	51	29	8	2	3	3		1		1	1	1	3.44	3.00	25
job-swp	44	23	39	33	86	14	3									3.61	4.00	36
ope-swp	30	22	45	73	33	25	10	1	1	2						3.81	4.00	28
tim-tim	28	6		2	8	83	44	30	29	3	4	4		1		6.33	6.00	23
mch-cas	31	7	11	7	8	26	55	50	23	11	1	3	3	4	2	6.51	7.00	25
ope-rev	21				2	36	36	44	91	3	1	3	4	1		7.49	8.00	21
ope-cas	14		1	2	7	16	54	60	49	14	10	3	1	3	8	7.92	8.00	13
job-rev	19	1		5	3	3	3	4	15	166	23					8.99	10.00	18
mch-ins	16		1	1	3	7	5	10	7	14	55	16	83	19	5	10.81	12.00	16
ope-adj	13			1					1	4	69	82	33	27	12	11.55	12.00	13
job-adj	13						1		1		50	80	51	30	16	11.83	12.00	13
mch-swp	15					1	1		5	7	6	29	27	102	49	12.69	14.00	15
mch-rev	15							1	1	3	10	12	20	43	137	13.36	15.00	15

Table 4: Tabu Search Ranks of Encoding-Neighborhood for 242 Classical Instances

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Mean	Median
job-ins	710	492	525	653	119	1										2.59	3.00
mch-cas	677	446	561	224	376	43	76	64	25	2	4	2				3.06	3.00
tim-tim	1164	244	179	205	285	110	107	99	59	39	6	3				3.06	2.00
ope-cas	253	738	341	410	259	155	169	124	32	8	5	4	2			3.72	3.00
job-swp	284	258	471	544	778	94	50	21								3.74	4.00
ope-ins	183	21	97	84	131	741	802	412	28		1					6.05	6.00
ope-swp	174	11	44	77	127	706	864	364	133							6.27	7.00
mch-adj	197	67	70	97	132	180	101	670	572	361	32	12	9			7.24	8.00
job-rev	17		4	44	90	378	157	571	722	508	9					8.11	8.00
ope-rev	43		2	2	10	27	26	89	853	1411	24	7	5	1		9.35	10.00
mch-ins	1						1	3	8	47	724	545	847	162	162	12.31	12.00
mch-rev									5	45	761	680	512	220	277	12.37	12.00
mch-swp							1	1	8	40	673	657	622	323	175	12.40	12.00
job-adj	2		1		1	1			4	6	256	296	192	942	799	13.66	14.00
ope-adj						1	1	1	2	11	255	294	205	942	788	13.66	14.00

Table 5: Tabu Search Ranks of Encoding-Neighborhood for 2500 Generated Instances

There are some common points. First for a fixed encoding, `ins` is generally better than `swp`, which is generally better than `rev`. This order is the same as the number of changes in the relative order of elements in the neighbor (1 for `ins`, 2 for `swp` and 1 to N for `rev`). Second, for a fixed neighborhood operator, `job` generally gives better results than `ope` which in turn generally gives better results than `mch`. The encodings are ordered like the complexity of the encodings, i.e. the number of encoded constraints.

Third common point, the worst group is the same by a large margin. In this group, we find `job-adj` and `ope-adj` which had nearly 100% neutrality and very large correlation length. We also find `mch-ins`, `mch-swp` and `mch-rev` that have many infeasible solutions, which is not a good feature for our implementation of tabu search, as explained before.

We observe some noticeable differences for the two sets of instances. First, the performances of `mch-cas` and `tim-tim` are very good for generated instances (mean rank of 3.06 for both) and average for classical instances (mean rank of 6.51 and 6.33 respectively). On the contrary, `mch-adj` is good for classical instances (mean rank of 3.11) and average for generated instances (mean rank of 7.24). We verified that the reason was not the random distribution used for the processing times of generated instances. In fact, the reason is the number of machines, which is never greater than 20 in classical instances and range from 10 to 100 in the generated instances.

Figure 5 shows the mean rank of each encoding-neighborhood pair with respect to the number of machines m and the number of jobs n for generated instances. In the first figure, we observe the three groups we identified. We also observe that for a low number of machines, the ranks can be very different from the ranks with a high number of machines ($m \geq 40$). In particular, `mch-cas` is in the two best pairs from $m = 30$ up to $m = 100$, but is only seventh for $m = 10$ and third for $m = 20$. `job-ins` is always good, which explains its overall result. `tim-tim` becomes the best pair from $m \geq 70$ and has globally a good behavior. In the second figure, we observe again the three groups. `mch-cas` shows a good behavior for a small number of jobs ($n \leq 50$) while `ope-cas` improves with increasing n. `job-ins` stays quite low, despite being outweighted by up to three other pairs between $n = 60$ and $n = 70$. Finally, `tim-tim` becomes the first pair for $n \geq 50$.

5. Conclusion

In our experiments, fitness landscape analysis fails at predicting the performance of the different encoding-neighborhood pairs. It is not possible to make a recommendation of encoding and neighborhood operator solely based

on the fitness landscape analysis result. Nevertheless, it can explain the bad performance when it exhibits extreme metrics, like very high neutrality. It implies that some pairs (`job-adj`, `ope-adj`, `mch-ins`, `mch-swp`, `mch-rev`) should be avoided for JSP. Regarding the performance of the encoding-neighborhood pairs with the tabu search algorithm, `job-ins`, `mch-cas` and `tim-tim` are the clear winners. Despite having different properties in their fitness landscape analysis, they share one similarity: they always stay in feasible solutions, either with the encoding that handle constraints when computing the schedule, or with the neighborhood that guarantees to provide a feasible solution. We aim at extending these results to more complex scheduling problems in order to see if these results can be extended or if new pairs will provide better results for different problems.

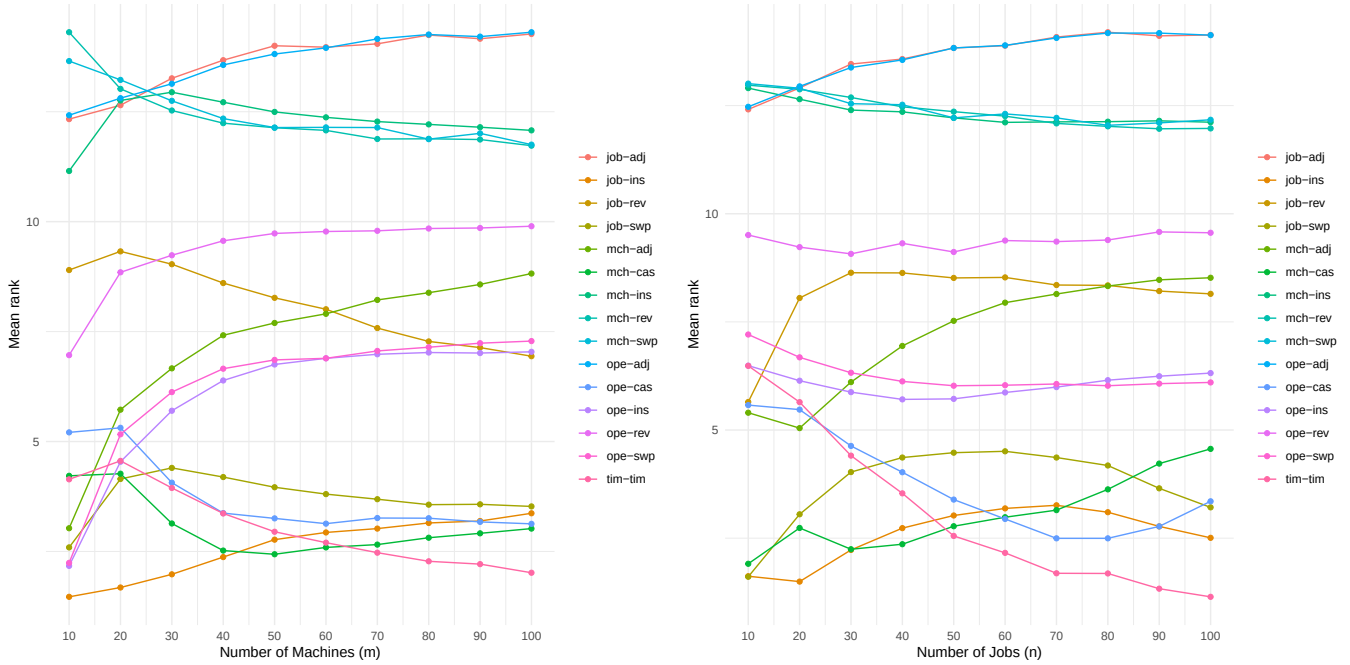

Figure 5: Mean Ranks of Encoding-Neighborhood w.r.t. Number of Machines (left) and Number of Jobs (right) for Generated Instances

References

Abdelmaguid, T. F. (2010). Representations in genetic algorithm for the job shop scheduling problem: A computational study. *Journal of Software Engineering and Applications*, *3*, 1155–1162. doi:10.4236/jsea.2010.312135.

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management science*, *34*, 391–401. doi:10.1287/mnsc.34.3.391.

Applegate, D., & Cook, W. (1991). A computational study of the job-shop scheduling problem. *ORSA Journal on computing*, *3*, 149–156. doi:10.1287/ijoc.3.2.149.

Bierwirth, C. (1995). A generalized permutation approach to job shop scheduling with genetic algorithms. *Operations-Research-Spektrum*, *17*, 87–92.

Bierwirth, C., Mattfeld, D., & Watson, J.-P. (2004). Landscape regularity and random walks for the job-shop scheduling problem. In J. Gottlieb, & G. Raidl (Eds.), *Evolutionary Computation in Combinatorial Optimization, 4th European Conference* (pp. 21–30). Springer-Verlag Berlin Heidelber. doi:10.1007/978-3-540-24652-7_3.

Cheng, R., Gen, M., & Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms—i. representation. *Computers & Industrial Engineering*, *30*, 983–997. doi:10.1016/0360-8352(96)00047-2.

Demirkol, E., Mehta, S., & Uzsoy, R. (1998). Benchmarks for shop scheduling problems. *European Journal of Operational Research*, *109*, 137–141. doi:10.1016/S0377-2217(97)00019-2.

Durasević, M., & Jakobović, D. (2016). Comparison of solution representations for scheduling in the unrelated machines environment. *39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, (pp. 1336–1342). doi:10.1109/MIPRO.2016.7522347.

Fernandez-Viagas, V., Perez-Gonzalez, P., & Framinan, J. M. (2019). Efficiency of the solution representations for the hybrid flow shop scheduling problem with makespan objective. *Computers & Operations Research*, *109*, 77–88. doi:10.1016/j.cor.2019.05.002.

Fisher, H. (1963). Probabilistic learning combinations of local job-shop scheduling rules. *Industrial scheduling*, (pp. 225–251).

Fisher, H., & Thompson, G. (1963). Probabilistic learning combinations of local job-shop scheduling rules. *Industrial scheduling*, (pp. 225–251).

Graham, R., Lawler, E., Lenstra, J., & Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In P. Hammer, E. Johnson, & B. Korte (Eds.), *Discrete Optimization II* (pp. 287–326). Elsevier volume 5 of *Annals of Discrete Mathematics*. URL: `https://www.sciencedirect.com/science/article/pii/S016750600870356X`. doi:https://doi.org/10.1016/S0167-5060(08)70356-X.

Hoorn, J. J. (2018). The current state of bounds on benchmark instances of the job-shop scheduling problem. *J. of Scheduling*, *21*, 127–128. doi:10.1007/s10951-017-0547-8.

Hoos, H. H., & Stützle, T. (2005). Search space structure and sls performance. In H. H. Hoos, & T. Stützle (Eds.), *Stochastic Local Search : Foundations and Applications* (pp. 203–253). Morgan Kaufmann. doi:10.1016/B978-155860872-6/50022-6.

Jorapur, V., Puranik, V. S., Deshpande, A. S., & Sharma, M. R. (2014). Comparative study of different representations in genetic algorithms for job shop scheduling problem. *Journal of Software Engineering and Applications*, *7*, 571–580. doi:10.4236/jsea.2014.77053.

Kuhpfahl, J., & Bierwirth, C. (2016). A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective. *Computers & Operations Research*, *66*, 44–57. doi:10.1016/j.cor.2015.07.011.

van Laarhoven, P. J. M., Aarts, E. H. L., & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations Research*, *40*, 113–125. doi:10.1287/opre.40.1.113.

Lawrence, S. (1984). Resource constrained project scheduling. *an experimental investigation of heuristic scheduling techniques*, .

Malan, K. (2021). A survey of advances in landscape analysis for optimisation. *Algorithms*, *14*, 40. doi:10.3390/a14020040.

Malan, K. M., & Engelbrecht, A. P. (2013). A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, *241*, 148–163. doi:10.1016/j.ins.2013.04.015.

Mastrolilli, M., & Gambardella, L. M. (2000). Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling*, *3*. doi:10.1002/(SICI)1099-1425(200001/02)3:1<3::AID-JOS32>3.0.CO;2-Y.

Mattfeld, D. C., Bierwirth, C., & Kopfer, H. (1999). A search space analysis of the job shop scheduling problem. *Annals of Operations Research*, *86*, 441–453. doi:10.1023/A:1018979424002.

Ponnambalam, S. G., Aravindan, P., & Rao, P. S. (2001). Comparative evaluation of genetic algorithms for job-shop scheduling. *Production Planning & Control*, *12*, 560–574. doi:10.1080/095372801750397680.

Storer, R. H., Wu, S. D., & Vaccari, R. (1992). New search spaces for sequencing problems with application to job shop scheduling. *Management science*, *38*, 1495–1509.

Strassl, S., & Musliu, N. (2022). Instance space analysis and algorithm selection for the job shop scheduling problem. *Computers & Operations Research*, *141*, 105661. doi:10.1016/j.cor.2021.105661.

Streeter, M. J., & Smith, S. F. (2006). How the landscape of random job shop scheduling instances depends on the ratio of jobs to machines. *Journal of Artificial Intelligence Research*, *26*, 247–287. doi:10.1613/jair.2013.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*, 278–285. doi:10.1016/0377-2217(93)90182-M.

Taillard, E. D. (1994). Parallel taboo search techniques for the job shop scheduling problem. *ORSA journal on Computing*, *6*, 108–117.

Tari, S., Basseur, M., & Goëffon, A. (2021). On the use of $(1, \lambda)$-evolution strategy as efficient local search mechanism for discrete optimization: a behavioral analysis. *Natural Computing*, *20*, 345–361. doi:10.1007/s11047-020-09822-2.

Tsujimura, Y., Gen, M., Cheng, R., & Momota, T. (1997). Comparative studies on encoding methods of ga for open shop scheduling. In *Australian Journal of Intelligent Information Processing Systems* (p. 214). volume 4.

Vlašic, I., Durasević, M., & Jakobović, D. (2020). A comparative study of solution representations for the unrelated machines environment. *Computers & Operations Research*, *123*, 105005. doi:10.1016/j.cor.2020.105005.

Yamada, T., & Nakano, R. (1992). A genetic algorithm applicable to large-scale job-shop problems. In *PPSN* (pp. 281–290). volume 2.

Şahman, M. A., & Korkmaz, S. (2022). Discrete artificial algae algorithm for solving job-shop scheduling problems. *Knowledge-Based Systems*, *256*, 109711. doi:10.1016/j.knosys.2022.109711.